VKUR
Your Ideas, Our Innovation

DON'T JUNK IT SELL IT

DESIGN SPECIFICATION DOCUMENT

VKUR
Your Ideas, Our Innovation

Prepared by

Kurra Sai Krishna
Chennuri Vidyadhari
Kanchukommula Uday Kumar
Kovvada Ravi Teja

Mentor
Prof Omar Aldawud

Contents

VKUR
Your Ideas, Our Innovation

VKUR
Your Ideas, Our Innovation

1. Introduction

Project Overview

The "Don't Junk It - List It" platform is a specialized online marketplace tailored for the buying and selling of junked car spare parts. It solves an unmet need in the market by providing a reliable platform where people can find or sell of used but still useful auto parts from the junked cars. This platform distinguishes itself by focusing exclusively on car parts, thereby catering to car enthusiasts, repair shops, and individual car owners. Its unique selling proposition lies in providing an easy-to-use, secure, and efficient platform for these specific transactions.

Document Purpose

The purpose of this Design Specification Document is to meticulously outline the technical framework for the "Don't Junk It - List It" project. It includes detailed descriptions of system architecture, user interface designs, component functionalities, database designs, security protocols, and other critical technical aspects. This document serves as a comprehensive guideline for software developers, UI/UX designers, database administrators, and project managers involved in the project. It is structured to facilitate a clear understanding of the project's technical requirements and to guide the development process towards achieving the set objectives with precision and efficiency.

Scope of the Document

The scope of this document encompasses:

- A detailed blueprint of the system architecture, distinguishing between the ReactJS-powered front-end for dynamic user interaction and the NodeJS with ExpressJS back-end for server-side data processing.
- An extensive design plan for user interfaces, tailored to the specific needs and workflows of admins, sellers, and customers, focusing on intuitive navigation and ease of use, all developed using ReactJS.
- A thorough explanation of each system component within the NodeJS framework, highlighting their individual functions, how they depend on and communicate with each other, ensuring seamless system integration.
- An in-depth structure of the database, detailing the schema, organization, and interrelations of critical data entities like user profiles, product listings, and transaction records.
- Integration of a comprehensive security framework, encompassing data protection strategies, JWT-based user authentication, and adherence to regulatory compliance standards.
- Plans for system performance and scalability, ensuring the system's efficiency and responsiveness are maintained as user demand increases, leveraging the strengths of NodeJS and database.
- A complete testing strategy, covering a variety of tests such as unit, integration, UI, and performance, to validate the reliability and functionality of each system component.
- Deployment methodology, focusing on effective deployment practices such as using Docker for containerization, which aids in streamlining deployment in cloud-based environments.
- Maintenance and support plans, outlining procedures for regular updates, issue resolution, and ensuring the system remains effective and relevant to user needs over time.

VKUR
Your Ideas, Our Innovation

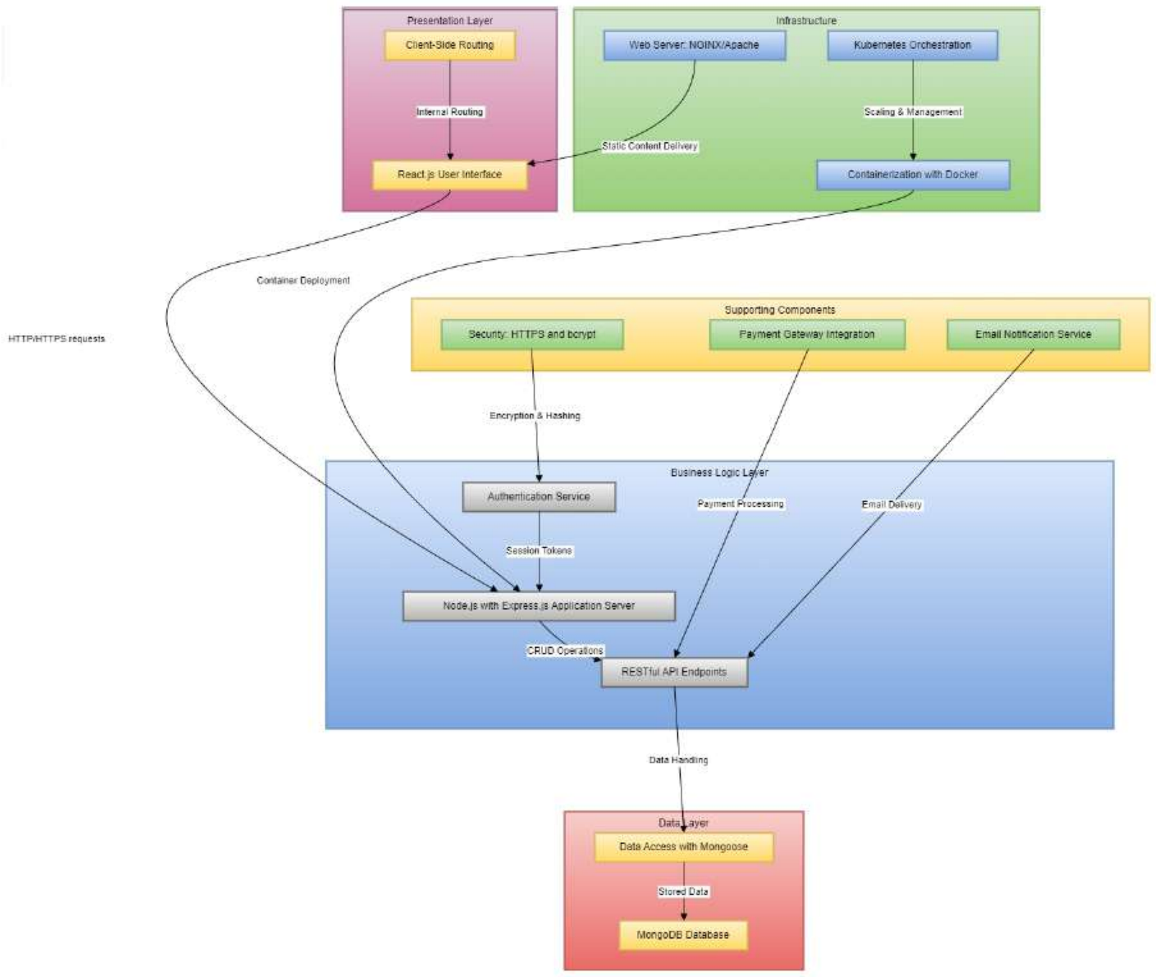
Confidential: © 2023 VKUR Innovations. All rights reserved.

2. System Overview

2.1 High-Level System Description

The "Don't Junk It - List It" platform is an e-commerce web application designed for the niche market of junked car spare parts. It operates on a three-tier architecture:

- **Front-End (Client-Side)**: Utilizes HTML, CSS, and JavaScript to provide an interactive user interface. The front-end is responsible for presenting information to users and collecting user input. It includes various pages such as Home, Product Listings, Cart, Orders, and User Profiles.
- **Back-End (Server-Side)**: Built on a server-side language Node.js, interfacing with a database for data storage. The back-end manages user authentication, processes business logic, and handles CRUD operations on products and orders.
- **Database**: MongoDB is used as the database management system, tasked with storing and organizing critical application data such as user profiles, product details, and transaction records.



2.2 Objectives and Goals

- **User-Friendly Interface**: Our aim is to make the platform really easy and pleasant to use. We want our users to find it simple to buy and sell car parts, with clear menus and helpful instructions.

Confidential: © 2023 VKUR Innovations. All rights reserved.

- **Efficient Marketplace**: We're creating a place where people can easily put up their car parts for sale and find the parts they need. The idea is to make the process of listing, searching, and buying as smooth as possible.
- **Secure Transactions**: It's super important to us that everyone's personal information and money are safe. So, we're putting strong security in place to protect our users when they're making transactions.
- **Scalability**: As more and more people use our platform, we want to make sure it still works great and doesn't slow down. We're planning for growth, ensuring the platform can handle more users and more listings over time.
- **Community Building**: We want to bring car lovers and experts together, making our platform a place where they love to come back to, share their knowledge, and connect with others who share their passion.

3. System Architecture

3.1 Architectural Design

The "Don't Junk It - List It" platform is developed using the MERN stack, embodying a three-tier architecture:

3.1.1 Front-End Service (Presentation Layer)

- Developed with React.js, this layer offers a dynamic and interactive user interface.
- It employs React's component-based architecture for modularity, facilitating quick development and updates.
- The front-end communicates with the server side via RESTful API calls, handling real-time data presentation and user interactions.

3.1.2 Authentication Service

- This service secures user sessions and data. It uses the OAuth protocol to authenticate users and manage sessions, ensuring that login processes meet industry-standard security practices.

VKUR Your Ideas, Our Innovation

3.1.3 Product Management Service

- A dedicated service for handling all operations related to car parts listings. It provides APIs for creating, reading, updating, and deleting (CRUD) listings, allowing sellers to manage their inventory through the platform.

3.1.4 Order Management Service

- This service oversees the entire order process, from initiation to fulfilment. It tracks the status of orders, manages order details, and interfaces with the payment service to confirm transactions.

3.1.5 Payment Service

- This component integrates with external payment providers like PayPal or Stripe. It handles all aspects of payment processing, including secure transmission of payment details, payment confirmation, and error handling.

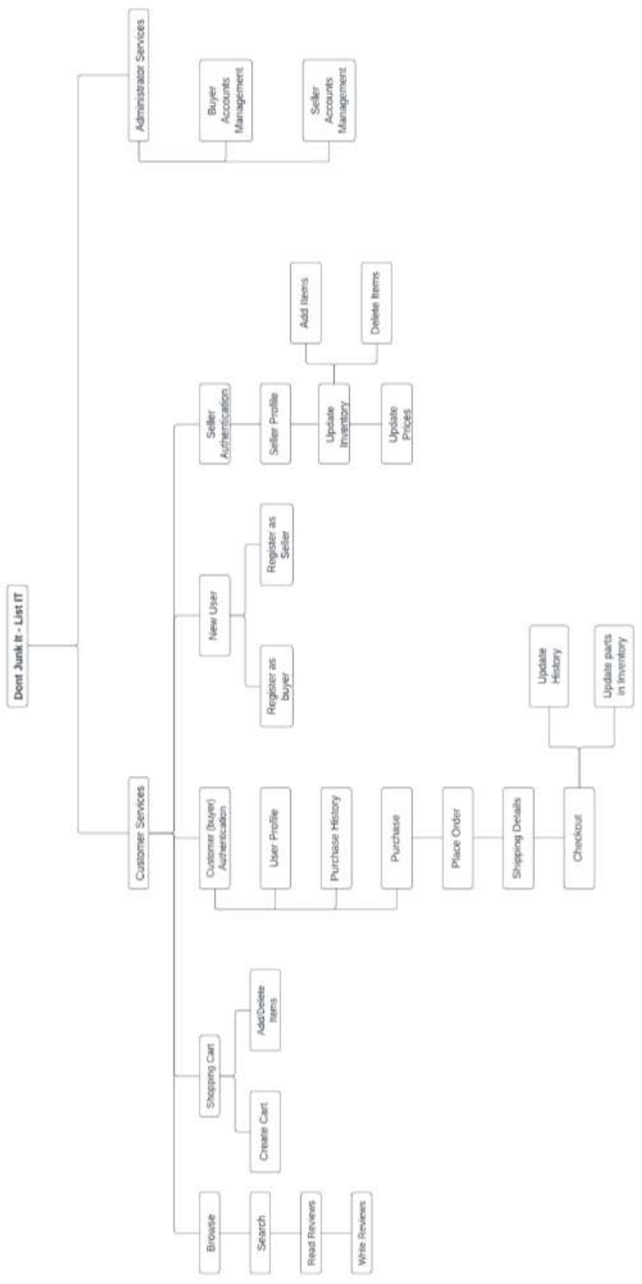
3.1.6 Notification Service

- An automated system that sends out email notifications to users. It uses transactional email services to send alerts for events such as order confirmations, shipping updates, and password resets.

Confidential: © 2023 VKUR Innovations. All rights reserved.

Each service within the "Don't Junk It - List It" platform is designed to intercommunicate through RESTful APIs, which follow the REST architecture's principles to enable stateless communication. This allows different services to exchange data in a structured format like JSON, promoting a clear and standardized interaction protocol.

We utilize Docker for containerization, which packages each service with its dependencies into isolated containers, making the application environment-agnostic and easily deployable. Kubernetes serves as the orchestration tool to manage these containers, handling the application's scaling and fault tolerance. This combination supports a continuous deployment pipeline, allowing us to deploy updates with minimal downtime.



Your Ideas, Our Innovation

3.2 System Components Overview

The system comprises key components:

- **Web Server**: This is the primary interface between the users and our system. It's responsible for delivering the React.js-based front-end to users' browsers. The server plays a crucial role in routing API calls from the front-end to our back-end services. For ensuring secure user interactions, it manages SSL termination, enabling HTTPS communications. We've also optimized it with caching strategies to speed up the loading of frequently accessed resources.
- **Back-End Server**: Developed with Node.js and Express.js, this server forms the backbone of our platform. It's here that we handle crucial operations like user authentication, where we use JSON Web Tokens (JWT) for maintaining secure sessions. The server is also responsible for all the heavy lifting related to product listings, like adding new items, updating existing ones, and removing unwanted listings. Another critical function is order processing; our server keeps track of customer orders from the moment they are placed until they are fulfilled. Additionally, it integrates with external payment services like PayPal, ensuring secure and smooth financial transactions.
- **Database (MongoDB)**: We chose MongoDB as our database solution because of its flexibility and efficiency in handling the varied data associated with e-commerce platforms. It stores all the vital data - from user profiles and product listings to detailed records of every transaction.
- **Load Balancer**: To make sure our server can handle traffic efficiently and to avoid any potential overloads or downtime, we use a load balancer. This tool smartly distributes incoming user requests across multiple servers, ensuring no single server gets overwhelmed. It also provides failover capabilities, which means if one server goes down, the traffic is automatically rerouted to other active servers, ensuring continuous availability of our platform.
- **External APIs**: Our platform integrates with various third-party services. For instance, we connect with payment gateways like PayPal and Stripe to handle transactions. We also use email service APIs for sending out automated emails, like order confirmations and marketing messages.

4. GUI Interfaces

4.1 User Interface Design

4.1.1 Admin Interface

- Dashboard showcasing system statistics (user signups, number of listings, sales figures).
- CRUD functionality for user accounts and product listings.
- Features for monitoring and managing orders.

4.1.2 Seller Interface

- Personal dashboard displaying their listings and sales history.
- Simple forms for listing new parts, including fields for part descriptions, images, and pricing.
- Overview of orders, including buyer details and order status.

4.1.3 Customer Interface

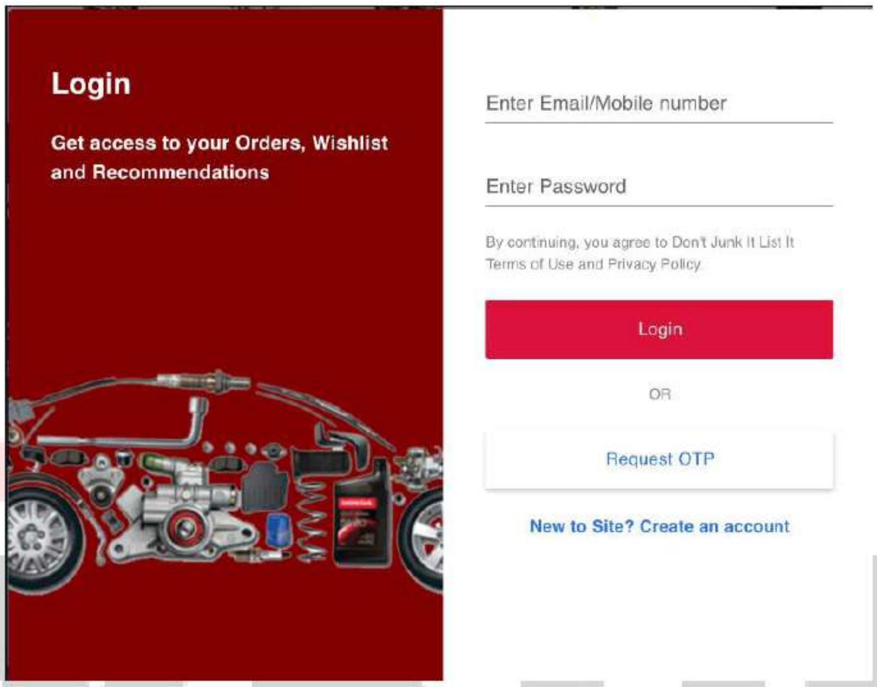
- Homepage featuring search functionality, category filters, and featured listings.
- Detailed product pages with high-quality images and part information.
- Easy-to-navigate cart and streamlined checkout process.

Confidential: © 2023 VKUR Innovations. All rights reserved.

4.2 Interface Layouts

✓ **Login and Signup Pages**:

- Clean layout with form fields for credentials.
- Links for password recovery and signup/login switch.



Login

Get access to your Orders, Wishlist and Recommendations

Enter Email/Mobile number

Enter Password

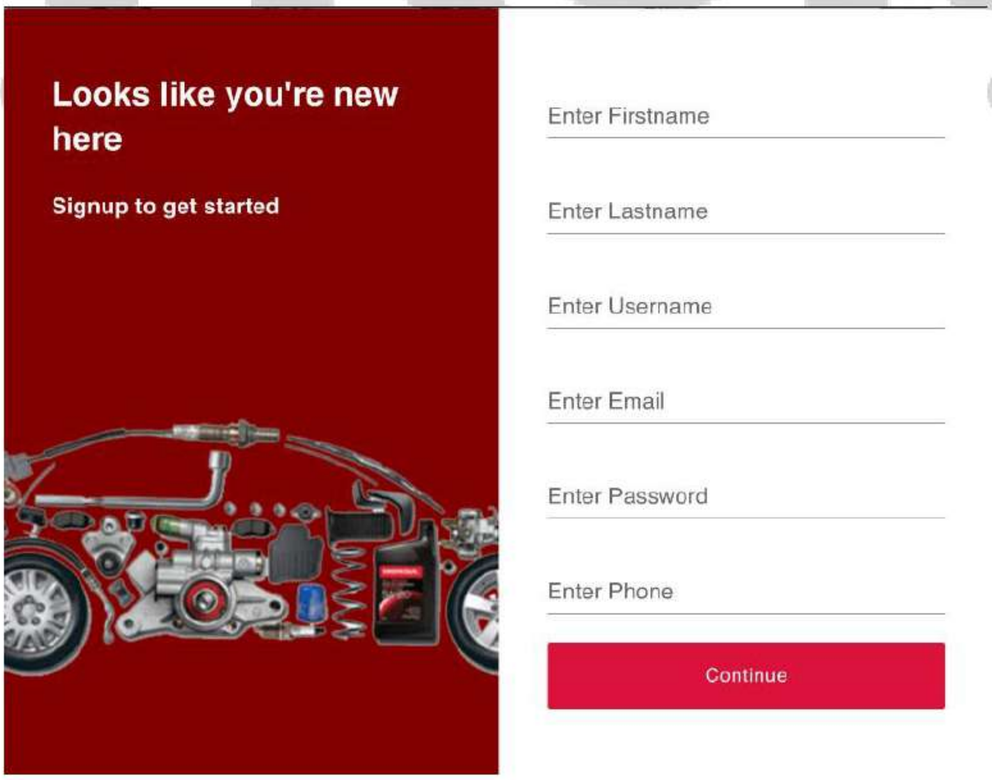
By continuing, you agree to Don't Junk It List It Terms of Use and Privacy Policy

Login

OR

Request OTP

New to Site? Create an account



Looks like you're new here

Signup to get started

Enter Firstname

Enter Lastname

Enter Username

Enter Email

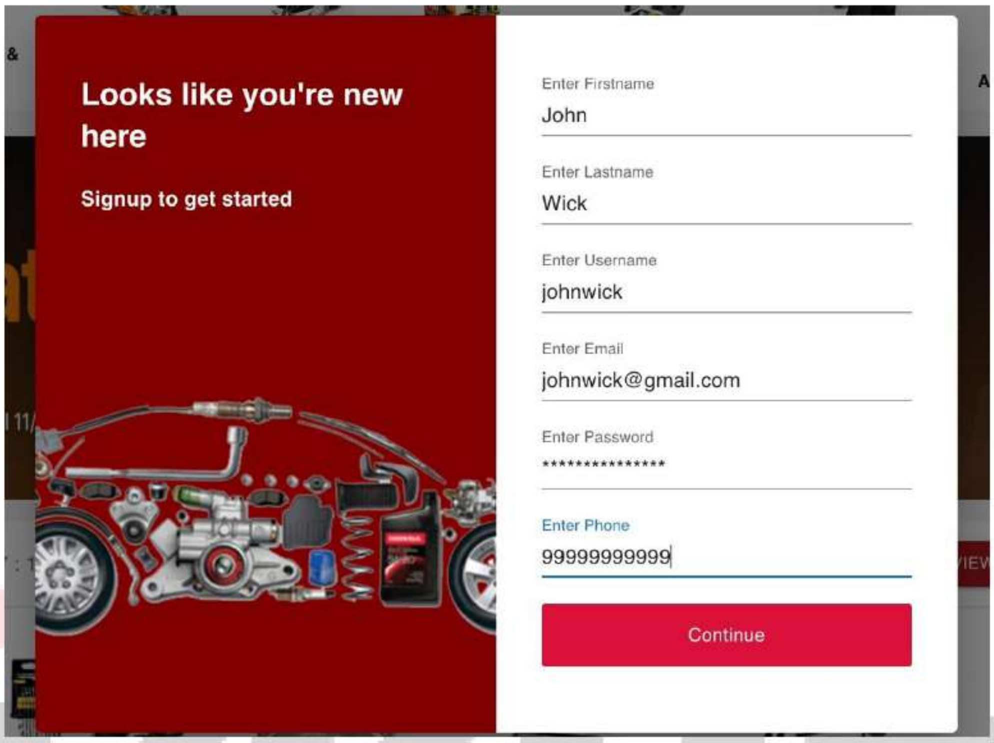
Enter Password

Enter Phone

Continue

VKUR Your Ideas, Our Innovation

Confidential: © 2023 VKUR Innovations. All rights reserved.



Enter Firstname
John

Enter Lastname
Wick

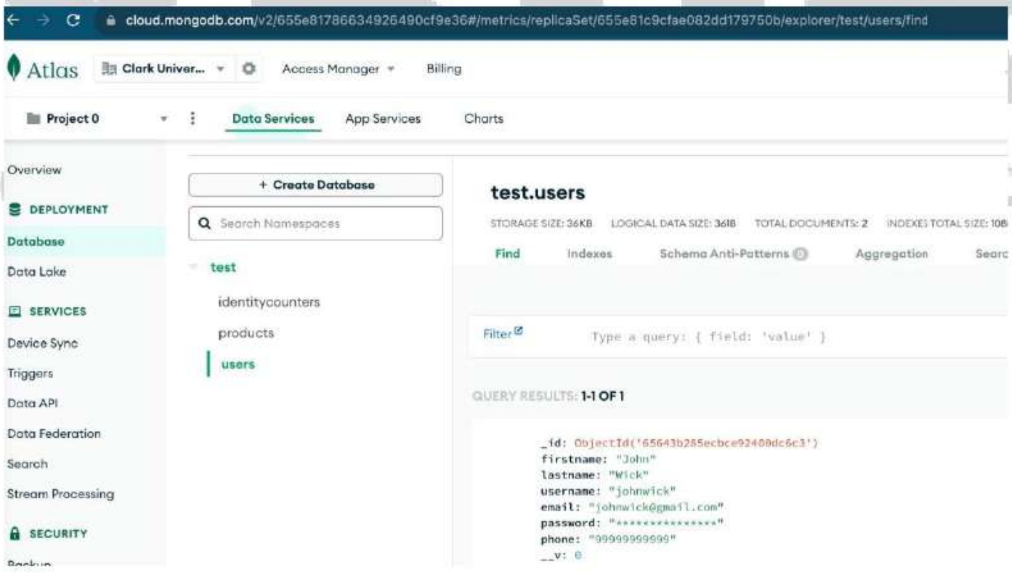
Enter Username
johnwick

Enter Email
johnwick@gmail.com

Enter Password
****

Enter Phone
99999999999

Upon registration, a MongoDB database entry is created.



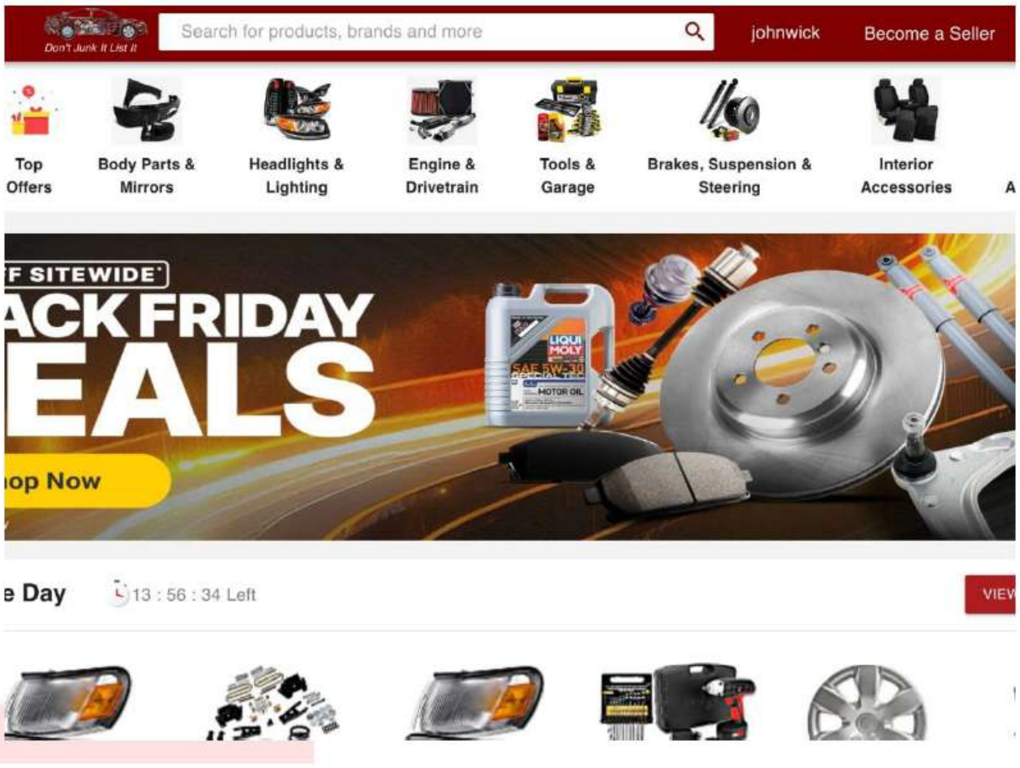
test.users

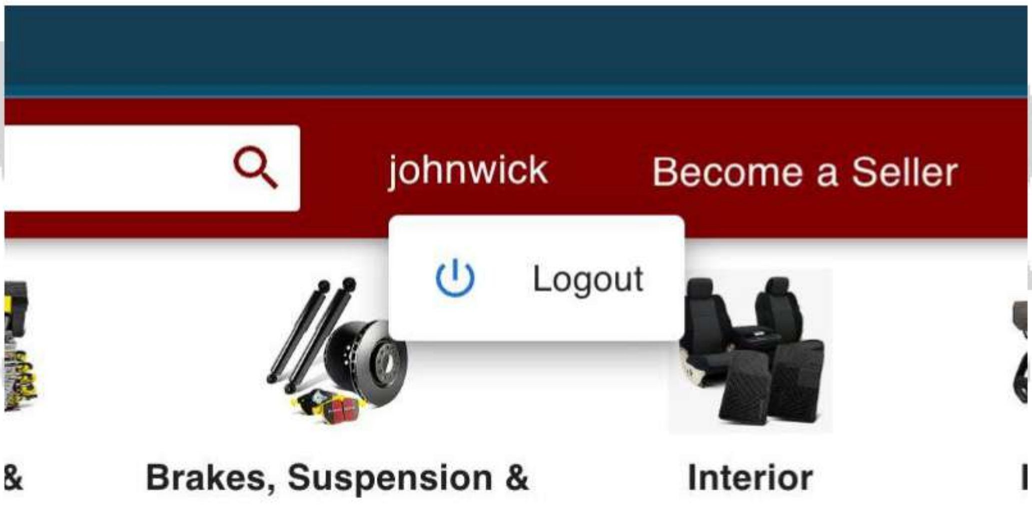
STORAGE SIZE: 36KB LOGICAL DATA SIZE: 361B TOTAL DOCUMENTS: 2 INDEXES TOTAL SIZE: 108

QUERY RESULTS: 1-1 OF 1

```
_id: ObjectId('65643b285ecbce92400dc6c3')
firstname: "John"
lastname: "Wick"
username: "johnwick"
email: "johnwick@gmail.com"
password: "***************"
phone: "99999999999"
__v: 0
```

Confidential: © 2023 VKUR Innovations. All rights reserved.

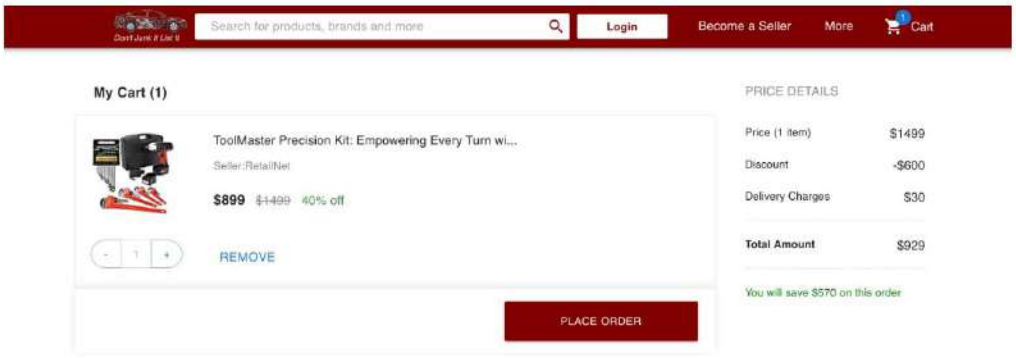




✓ **Cart Page**:

- List items in the cart with options to update quantity or remove items.
- Display total price and checkout button.

Confidential: © 2023 VKUR Innovations. All rights reserved.


Search for products, brands and more

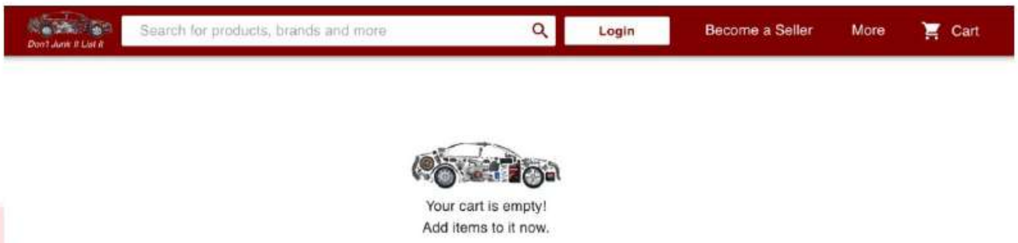
My Cart (1)

ToolMaster Precision Kit: Empowering Every Turn wi...
Seller:RetailNet
$899 ~~$1499~~ 40% off

PRICE DETAILS

Price (1 item)	$1499
Discount	-$600
Delivery Charges	$30
Total Amount	$929

You will save $570 on this order


Search for products, brands and more

Your cart is empty!
Add items to it now.

5. Component Design

5.1 User Authentication Component

The User Authentication Component is designed with a focus on security and user experience

5.1.1 Design and Workflow

- When a user signs in, the component verifies their credentials against stored data in the MongoDB database.
- Upon successful authentication, a JWT is generated, which is used for maintaining the user's session. This token is stateless, meaning it doesn't require server-side storage and can be easily scaled.

5.1.2 Security Measures

- Passwords are hashed using bcrypt before storing in the database, providing robust protection against brute-force attacks.
- The use of HTTPS ensures that all communication between the client and server is encrypted.

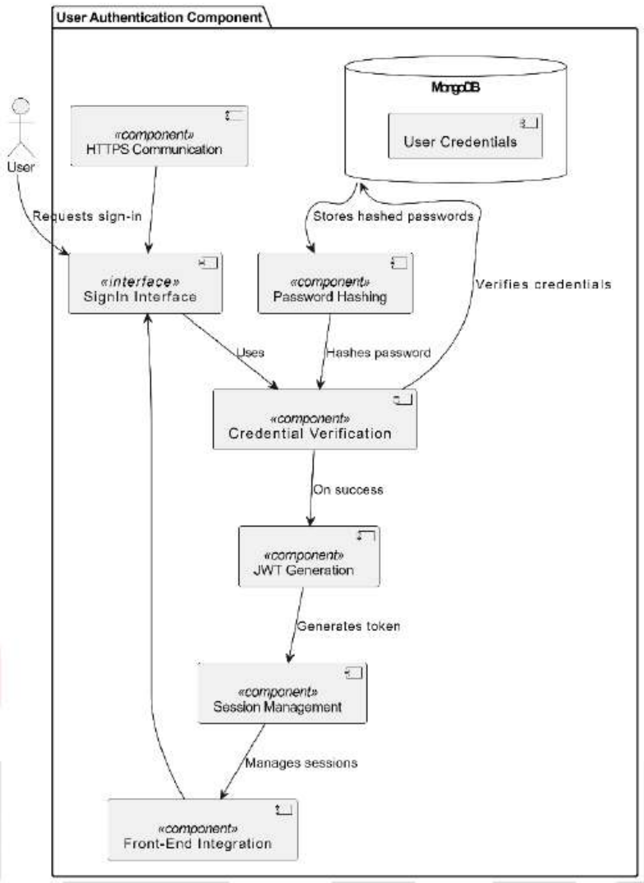
5.1.3 Integration with Front-End

- The authentication process is integrated with the ReactJS front-end, providing users with feedback and prompts as necessary, like error messages for incorrect credentials.

5.1.4 Session Management

- JWTs are used to manage user sessions. Once logged in, the user's interactions are authenticated using the token, which contains encoded user information and an expiration time.

Confidential: © 2023 VKUR Innovations. All rights reserved.


User Authentication Component
MongoDB
User Credentials
User
«component» HTTPS Communication
Requests sign-in
Stores hashed passwords
«interface» SignIn Interface
«component» Password Hashing
Verifies credentials
Uses
Hashes password
«component» Credential Verification
On success
«component» JWT Generation
Generates token
«component» Session Management
Manages sessions
«component» Front-End Integration

This component ensures that only authorized users can access and interact with the platform, maintaining the integrity and security of user data and transactions.

5.2 Product (Parts) Management Component

The Product (Parts) Management Component is a crucial element designed for a seamless and effective management of car part listings

5.2.1 User Interaction

- Sellers use this component to list car parts for sale. They can enter detailed information like part name, description, category, price, and upload images.
- This information is entered through a user-friendly interface developed in ReactJS, ensuring ease of use.

5.2.2 Database Management

- The component interacts with MongoDB, our chosen database. It stores detailed listings and manages them efficiently. Whenever a seller adds or modifies a listing, the changes are reflected in the database in real-time.

5.2.3 Search and Filtering Capabilities

- One of the key features is the advanced search and filtering options available to buyers.
- Buyers can search for specific parts using various filters like category, price range, or condition. This functionality uses MongoDB's querying capabilities to fetch and display relevant listings.

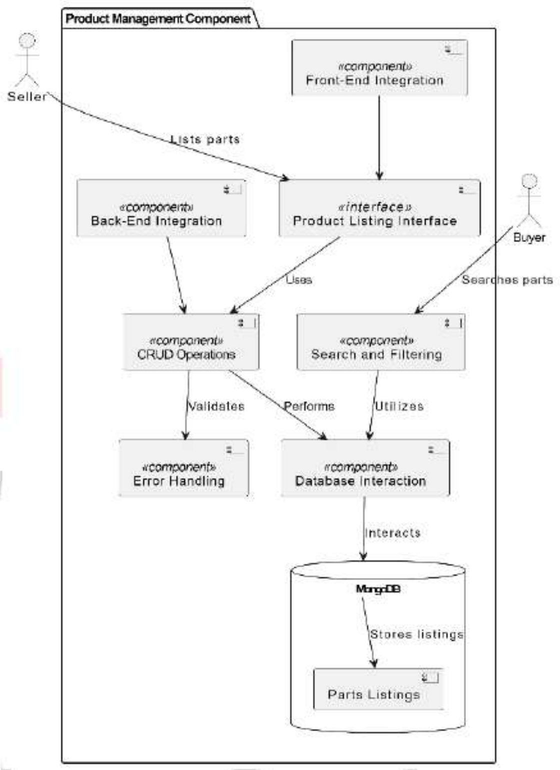
Confidential: © 2023 VKUR Innovations. All rights reserved.

5.2.4 CRUD Operations

- The component supports all CRUD (Create, Read, Update, and Delete) operations. Sellers can create new listings, view their existing listings, update details, or remove listings as needed.
- Robust error handling and validation are implemented to ensure data integrity.

5.2.5 Integration with Front-End and Back-End

- It's tightly integrated with the ReactJS front-end for a smooth user experience and with the Node.js back-end for processing and business logic.


Product Management Component
Seller
«component» Front-End Integration
Lists parts
«component» Back-End Integration
«interface» Product Listing Interface
Buyer
Uses
Searches parts
«component» CRUD Operations
«component» Search and Filtering
Validates
Performs
Utilizes
«component» Error Handling
«component» Database Interaction
Interacts
MongoDB
Stores listings
Parts Listings

Your Ideas, Our Innovation

Overall, this component ensures that managing car part listings is efficient and straightforward for sellers, while also making it easy for buyers to find exactly what they need.

5.3 Order Processing Component

The Order Processing Component in "Don't Junk It - List It" is designed to efficiently handle the lifecycle of orders from initiation to completion:

5.3.1 Order Placement and Tracking

- When a customer places an order, this component records it, capturing details like buyer information, ordered parts, quantities, and prices.
- It updates order statuses through different stages such as 'placed', 'processing', 'shipped', and 'delivered'.

5.3.2 Integration with Inventory

- It interacts with the Product Management Component to adjust inventory based on orders, ensuring accurate stock levels.

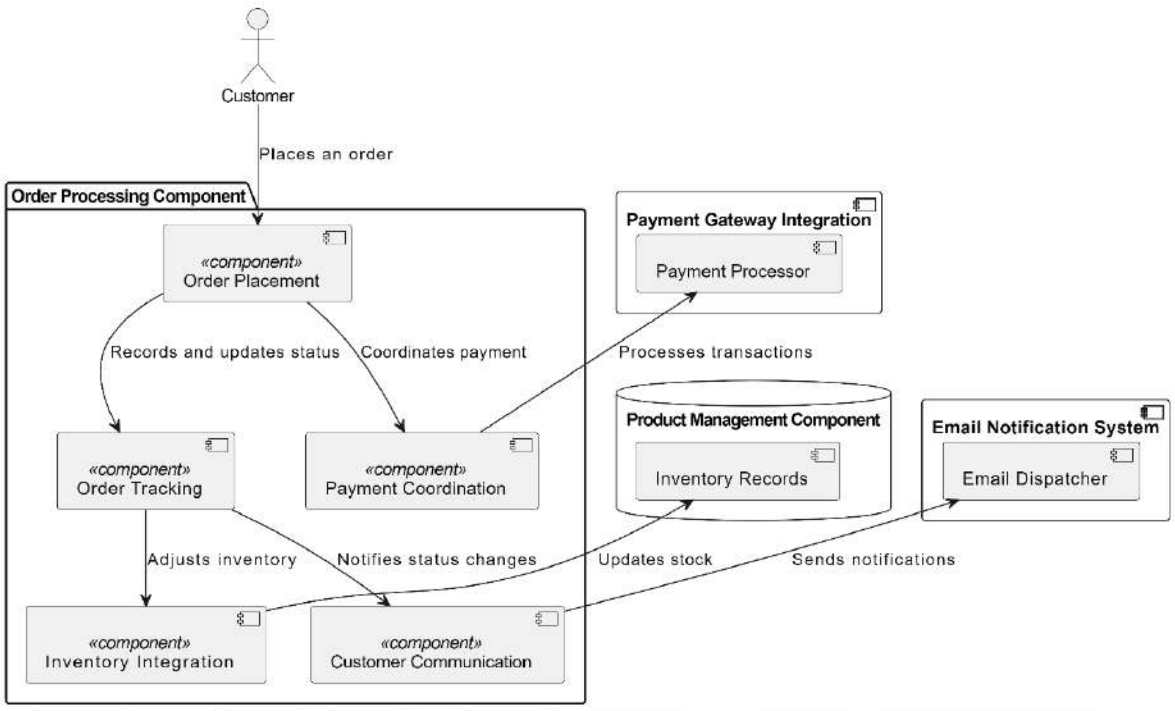
5.3.3 Customer Communication

- Facilitates communication with customers about their order status, leveraging the Email Notification System for updates.

Confidential: © 2023 VKUR Innovations. All rights reserved.

5.3.4 Payment Processing Coordination

- Works in tandem with the Payment Gateway Integration Component to manage financial transactions related to orders.


Customer
Places an order
Order Processing Component
«component» Order Placement
Records and updates status
Coordinates payment
«component» Order Tracking
«component» Payment Coordination
Adjusts inventory
Notifies status changes
«component» Inventory Integration
«component» Customer Communication
Payment Gateway Integration
Payment Processor
Processes transactions
Product Management Component
Inventory Records
Email Notification System
Email Dispatcher
Updates stock
Sends notifications

This component streamlines the order process, ensuring a smooth and reliable experience for both buyers and sellers.

5.4 Payment Gateway Integration

The Payment Gateway Integration Component in "Don't Junk It - List It" has been developed with meticulous attention to security and user experience:

5.4.1 Robust Integration with Payment Services

- The component is integrated with leading payment services like PayPal and Stripe, offering users a range of secure payment options.
- This integration is handled through secure APIs, ensuring that sensitive payment data is encrypted and securely processed.

5.4.2 Seamless Transaction Processing

- It manages the complexities of online transactions, including payment authorization, capturing, and processing.
- The component verifies transaction details and ensures funds are correctly transferred.

5.4.3 Synchronization with Order Processing

- The payment status directly influences the order processing workflow. Successful payments prompt the Order Processing Component to update the order status to 'paid' and proceed with fulfilment.
- In cases of failed transactions, it communicates this to the user and the order remains in a pending state.

VKUR Your Ideas, Our Innovation

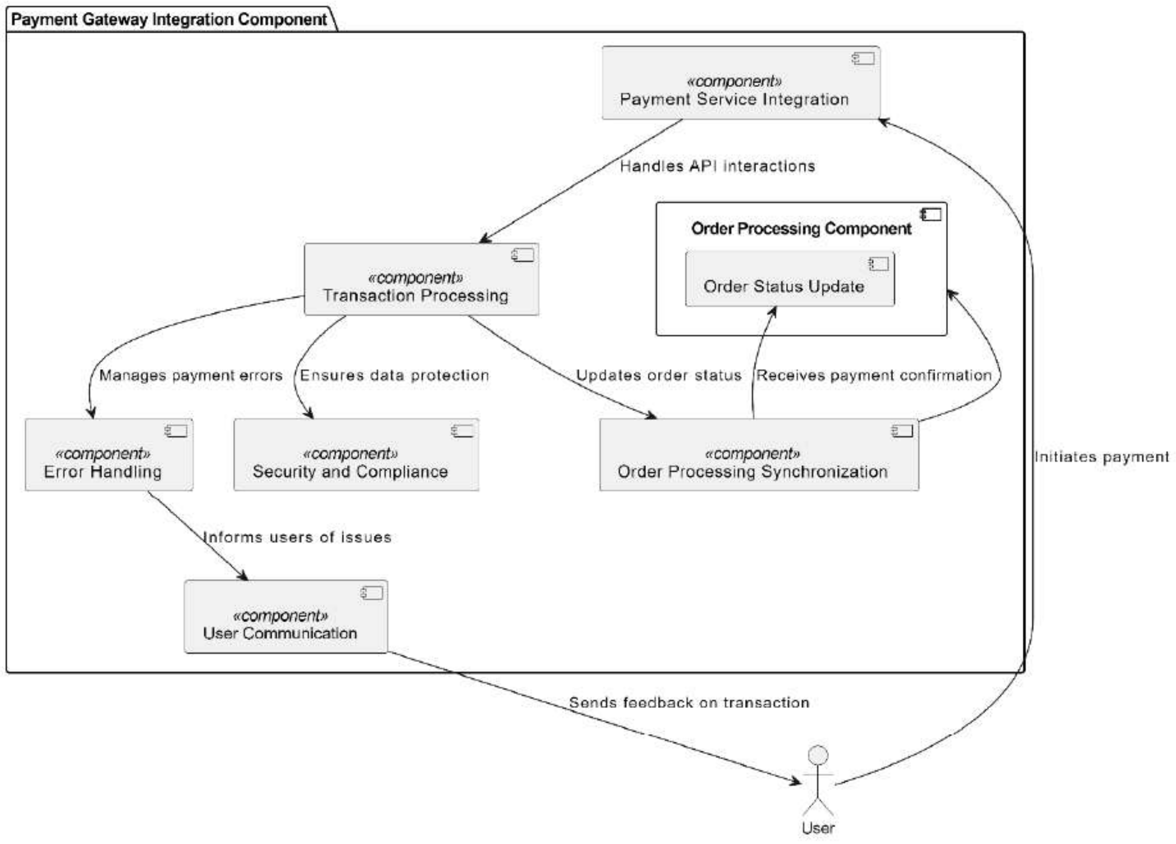
Confidential: © 2023 VKUR Innovations. All rights reserved.

5.4.4 Security and Compliance

- Adherence to security standards like PCI DSS is a priority, ensuring the safeguarding of user financial information.
- Regular updates and security checks are part of the component's maintenance to stay compliant with evolving security standards.

5.4.5 Error Handling and User Communication

- The component is equipped with comprehensive error handling mechanisms to address any issues during the payment process.
- It provides clear, understandable feedback to the user in case of transaction failures or issues, enhancing the user experience.


Payment Gateway Integration Component
«component» Payment Service Integration
Handles API interactions
Order Processing Component
Order Status Update
«component» Transaction Processing
Manages payment errors
Ensures data protection
Updates order status
Receives payment confirmation
«component» Error Handling
«component» Security and Compliance
«component» Order Processing Synchronization
Initiates payment
Informs users of issues
«component» User Communication
Sends feedback on transaction
User

This component is crucial in ensuring that financial transactions on the platform are smooth, secure, and user-friendly, contributing significantly to the trust and reliability of the "Don't Junk It - List It" platform.

5.5 Email Notification System

The Email Notification System in "Don't Junk It - List It" is intricately designed for efficient and effective user communication:

5.5.1 Automated Email Triggers

- The system is set up to automatically dispatch emails in response to various user actions and system events. For instance, when an order is placed, confirmed, or shipped, the system sends timely notifications to the relevant customers.

Confidential: © 2023 VKUR Innovations. All rights reserved.

5.5.2 Seamless Integration with Other Components

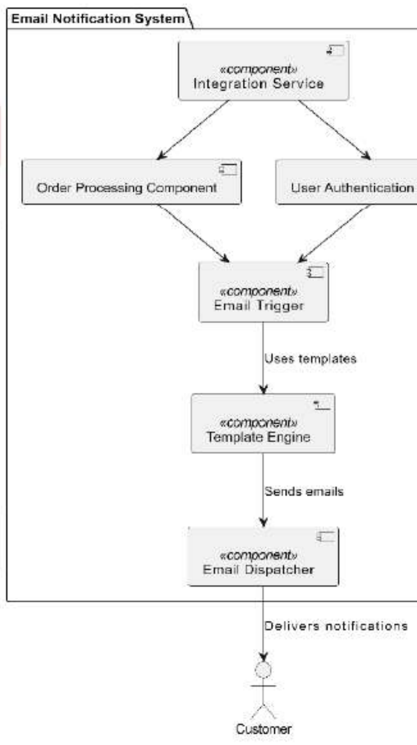
- This system works in tandem with the Order Processing and User Authentication Components. Any significant updates or changes in these areas, such as order status updates or password reset confirmations, prompt the Email System to send corresponding notifications.

5.5.3 Customizable and Consistent Email Templates

- We use a series of predefined, customizable templates for different types of emails. These templates ensure consistency in branding and messaging. They are designed to be clear and informative, enhancing the user experience by providing relevant information in an easily digestible format.

5.5.4 Scalability and Reliability

- The system is engineered to handle a high volume of emails efficiently. It's scalable to accommodate the growing user base and activity on the platform, ensuring that no user misses out on important communications.


Email Notification System
«component» Integration Service
Order Processing Component
User Authentication Component
«component» Email Trigger
Uses templates
«component» Template Engine
Sends emails
«component» Email Dispatcher
Delivers notifications
Customer

This Email Notification System is a vital component of the platform, fostering user engagement and trust through consistent, timely, and relevant communications.

6. Database Design

6.1 Database Schema and Structure

Given the system being an online marketplace for used automotive parts, the Logical Data Model (LDM) could comprise entities such as Users, Automotive Parts, Orders, Reviews, and Content Pages.

Table Name	Description
Customer	Stores information about customers, including personal details and login credentials.
Card Details	Contains credit card information linked to customers for transactions.

Confidential: © 2023 VKUR Innovations. All rights reserved.

Order Details	Records the details of each order, including shipping information and purchase dates.
Cart	Temporarily holds items that customers intend to purchase, along with price and quantity.
Parts	Lists the car parts available for sale, including name, price, and stock availability.
Parts Review	Captures customer reviews for parts, including ratings and review text.
Shipping Type	Details the shipping methods available with associated cost and delivery times.
Purchase History	Logs the history of purchases made by customers, including the parts bought and their prices.
Tax	Details the tax rates applicable to transactions, based on state.

Relationships:

- **Customer to Card Details**: A one-to-one relationship, indicating that each customer has a single set of card details associated with their account for transactions.
- **Customer to Order Details**: A one-to-many relationship, where a single customer can have multiple orders recorded in the system.
- **Customer to Parts Review**: A one-to-many relationship, where a customer can write multiple reviews for different parts.
- **Order Details to Cart**: A one-to-many relationship, as each order can contain multiple cart items, but each cart item is associated with one order.
- **Cart to Parts**: A many-to-one relationship, meaning multiple cart entries can refer to a single part if multiple customers select the same part.
- **Order Details to Shipping Type**: A many-to-one relationship, where multiple orders can be shipped using the same shipping type.
- **Order Details to Purchase History**: A one-to-many relationship, as each order contributes to a single entry in the purchase history per customer.
- **Order Details to Tax**: A many-to-one relationship, suggesting that orders from different states can have the same tax rate applied if the state is the same.

VKUR
Your Ideas, Our Innovation

Customer Table

Field	Data Type	Description
UserID	Varchar	Unique identifier for the customer
FName	Varchar	Customer's first name
LName	Varchar	Customer's last name
Email	Varchar	Customer's email address
Contact	Integer	Customer's contact number

Card Details Table

Field	Data Type	Description
CardID	Varchar	Unique identifier for the card details
CardHolderName	Varchar	Name of the cardholder
CardNumber	Varchar	Credit card number

Confidential: © 2023 VKUR Innovations. All rights reserved.

CVV	Integer	Card verification value
ExpiryDate	Date	Expiration date of the card
UserID	Varchar	Foreign key from Customer table

Order Details Table

Field	**Data Type**	**Description**
OrderID	Integer	Unique identifier for the order
UserID	Varchar	Foreign key from Customer table
Receiver's Name	Varchar	Name of the order receiver
Address	Varchar	Shipping address for the order
City	Varchar	City for the shipping address
Zip	Integer	Zip code for the shipping address
State	Varchar	State for the shipping address
ShippingType	Varchar	Foreign key from Shipping Type table
DateOfPurchase	Date	Date when the purchase was made

Cart Table

Field	**Data Type**	**Description**
CartID	Integer	Unique identifier for the cart item
PartID	Integer	Foreign key from Parts table
Price	Double	Price of the part in the cart
UserID	Varchar	Foreign key from Customer table
Date	Date	Date when the item was added to the cart
Quantity	Integer	Quantity of the part in the cart

VKUR
Your Ideas, Our Innovation

Parts Table

Field	**Data Type**	**Description**
PartID	Integer	Unique identifier for the car part
PName	Varchar	Name of the car part
Price	Double	Price of the car part
Availability	Integer	Availability stock count of the car part

Parts Review Table

Field	**Data Type**	**Description**
PartID	Integer	Unique identifier for the car part reviewed
Reviews	Varchar	Review text content
Rating	Varchar	Rating given by the customer
ReviewDate	Date	Date when the review was posted
UserName	Varchar	Username of the customer who posted the review

Shipping Type Table

Confidential: © 2023 VKUR Innovations. All rights reserved.

Field	Data Type	Description
ShippingType	Varchar	Description of the shipping method
Price	Double	Cost associated with the shipping method
DaysForDelivery	Integer	Estimated delivery time in days for the shipping method

Purchase History Table

Field	Data Type	Description
UserID	Varchar	Foreign key from Customer table
PartID	Integer	Foreign key from Parts table
OrderID	Integer	Foreign key from Order Details table
DateOfPurchase	Date	Date when the part was purchased
Quantity	Integer	Quantity of parts purchased
Price	Double	Total price paid for the parts

Tax Table

Field	Data Type	Description
StateName	Varchar	The name of the state
TaxRate	Double	Applicable tax rate for the state
StateID	Integer	Unique identifier for the state (PK)

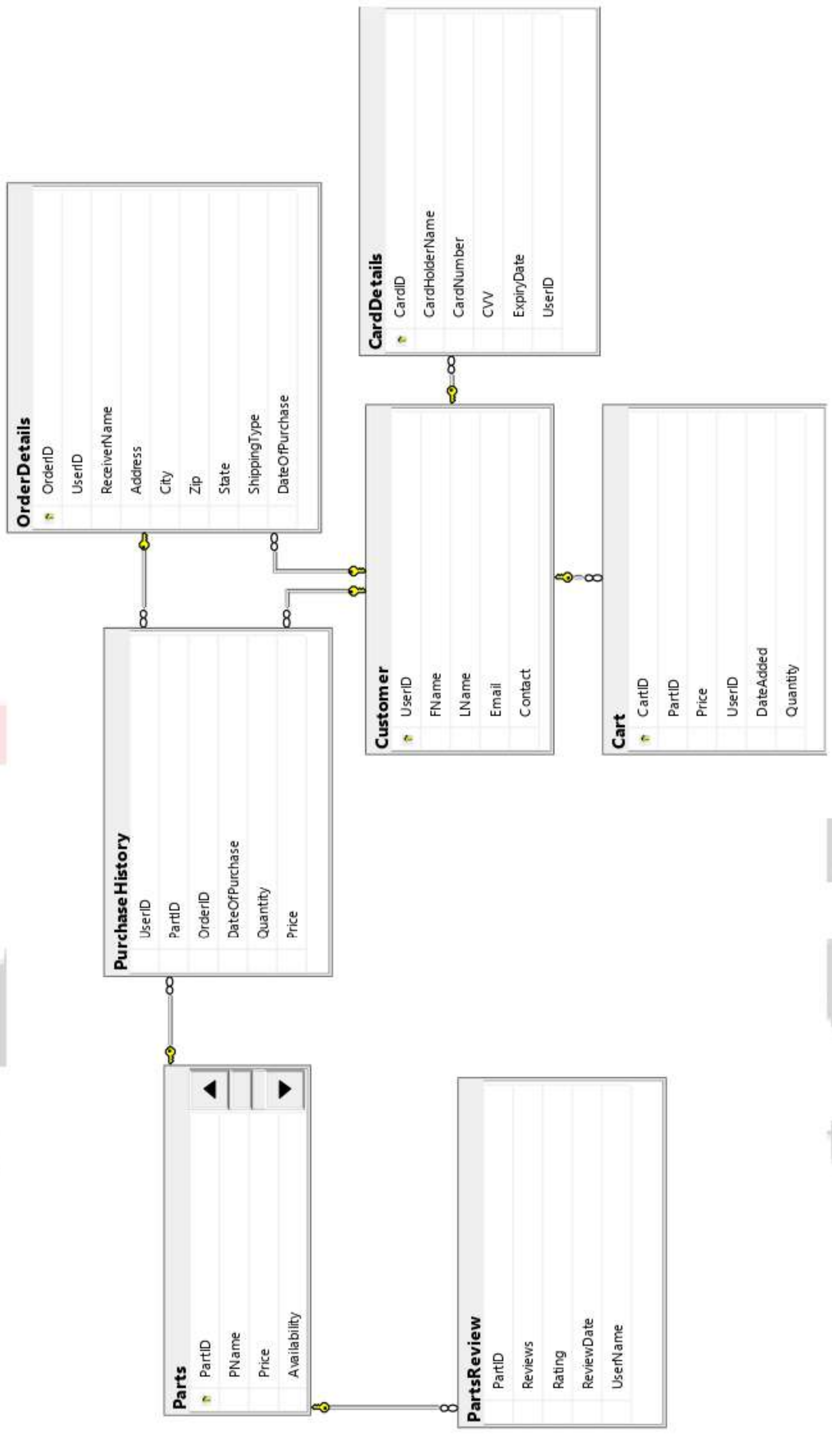
VKUR
Your Ideas, Our Innovation

6.2 Entity-Relationship Diagrams

6.2.1 Physical schema

Represents the actual structure of the database, including tables, columns, data types, and constraints like primary keys and foreign keys. The physical schema will detail indexes, triggers, and other performance-related database structures.

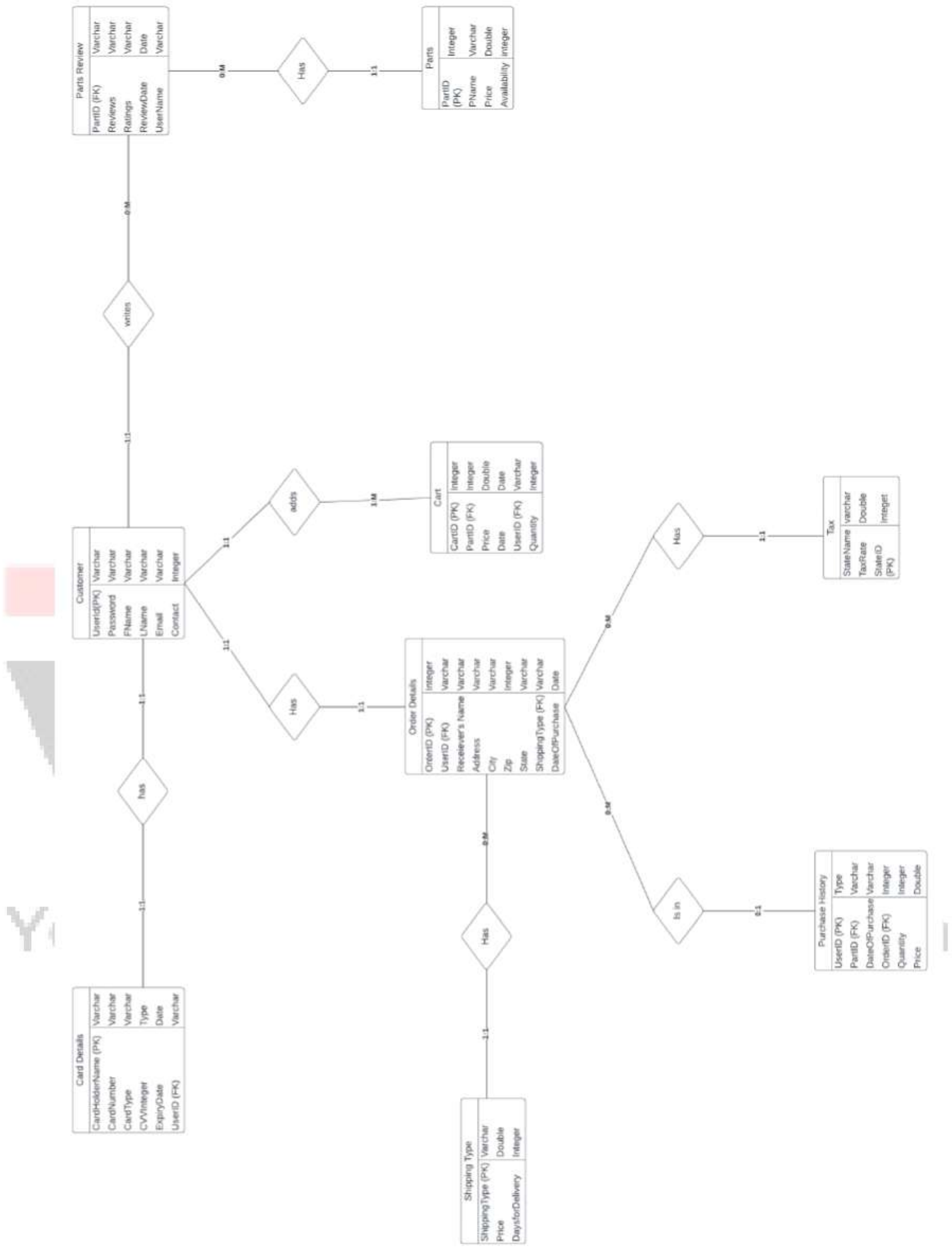
Confidential: © 2023 VKUR Innovations. All rights reserved.



6.2.2 Logical schema

Showcases the entities, their attributes, and the relationships between them without getting into the details of how they are implemented in the actual database system. It is more abstract and focuses on the business rules and the general data model. The logical schema is concerned with how the system should work from a data perspective.

Confidential: © 2023 VKUR Innovations. All rights reserved.



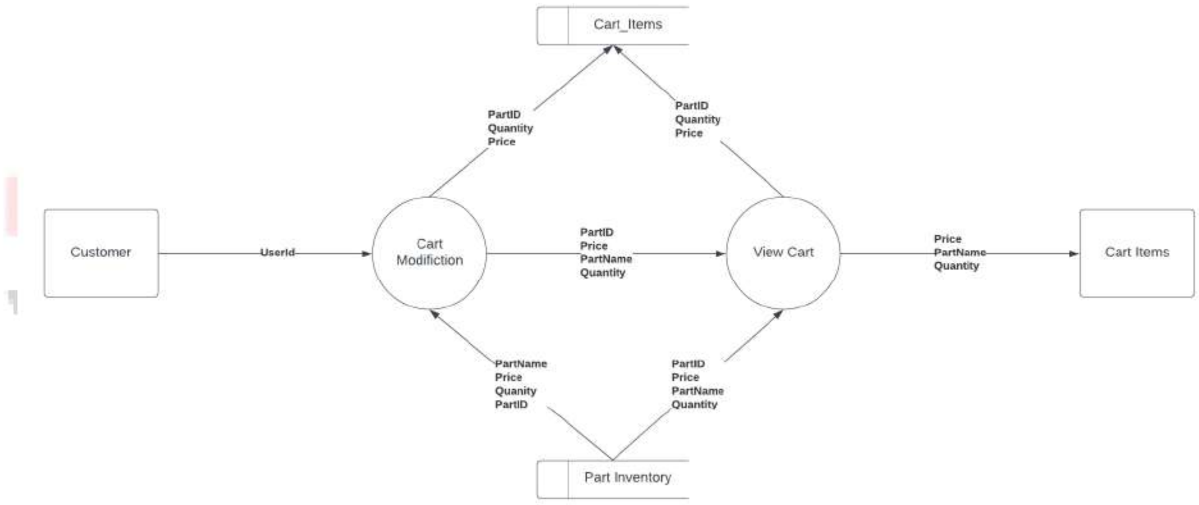
6.3 Data Flow Diagrams

6.3.1 Shopping cart system

Data Flow Diagram (DFD) for a shopping cart system is focused on the flow of information between the customer and the cart items within the system. Here's a detailed explanation

Confidential: © 2023 VKUR Innovations. All rights reserved.

- **Customer**: This entity represents the users of the platform, who have the capability to interact with their shopping cart.
- **Cart Modification Process**: This process shows how customers can modify their cart, likely adding or removing items. It details the interaction between the customer and the cart items, including part IDs, quantities, and prices.
- **View Cart Process**: This process allows customers to view the contents of their cart. It connects the customer with the current items in their cart, displaying part IDs, names, prices, and quantities.
- **Cart Items Entity**: This is likely a database table or a data store that keeps track of all the items currently in the customer's cart. It includes details like part IDs, quantities, and prices.
- **Part Inventory Entity**: This represents the inventory system or database where all available parts are stored. It interacts with the cart modification process to update the inventory based on customer actions.



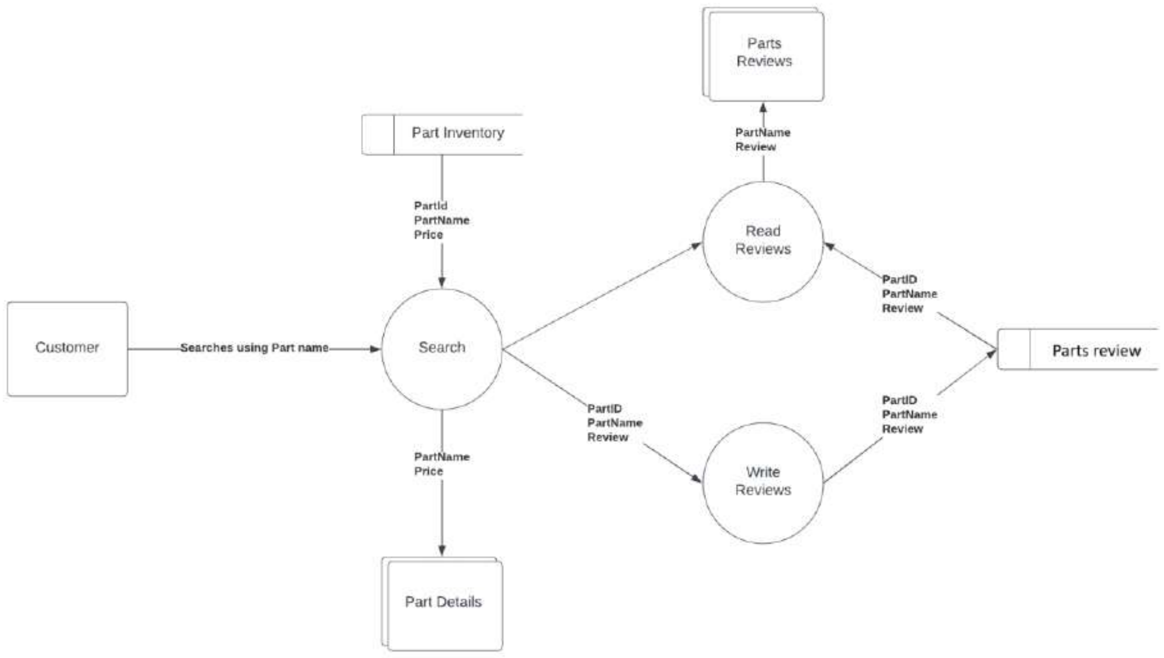
6.3.2 Car parts search and review functionalities

DFD demonstrates how customers interact with the system to search for parts, view part details, read existing reviews, and write new reviews.

Your Ideas. Our Innovation

- **Customer**: This entity represents the users of the platform who are looking to interact with the parts inventory.
- **Search**: The central process where customers search for parts using specific names. This process interacts with the "Part Inventory" data store to retrieve part details like ID, name, and price.
- **Part Inventory**: This data store contains all the information about available parts.
- **Part Details**: The resulting details from the search are shown here, including part name and price.
- **Read Reviews**: Customers can read reviews for parts. This process retrieves information such as part name and review content from the "Parts Reviews" data store.
- **Parts Reviews**: This data store holds the reviews that customers have written for parts.
- **Write Reviews**: This process allows customers to write reviews for parts. This new review data is then stored in the "Parts review" data store.
- **Parts review**: This data store collects new reviews written by customers, containing the part ID, part name, and the review content.

Confidential: © 2023 VKUR Innovations. All rights reserved.



6.3.3 Customer purchase process

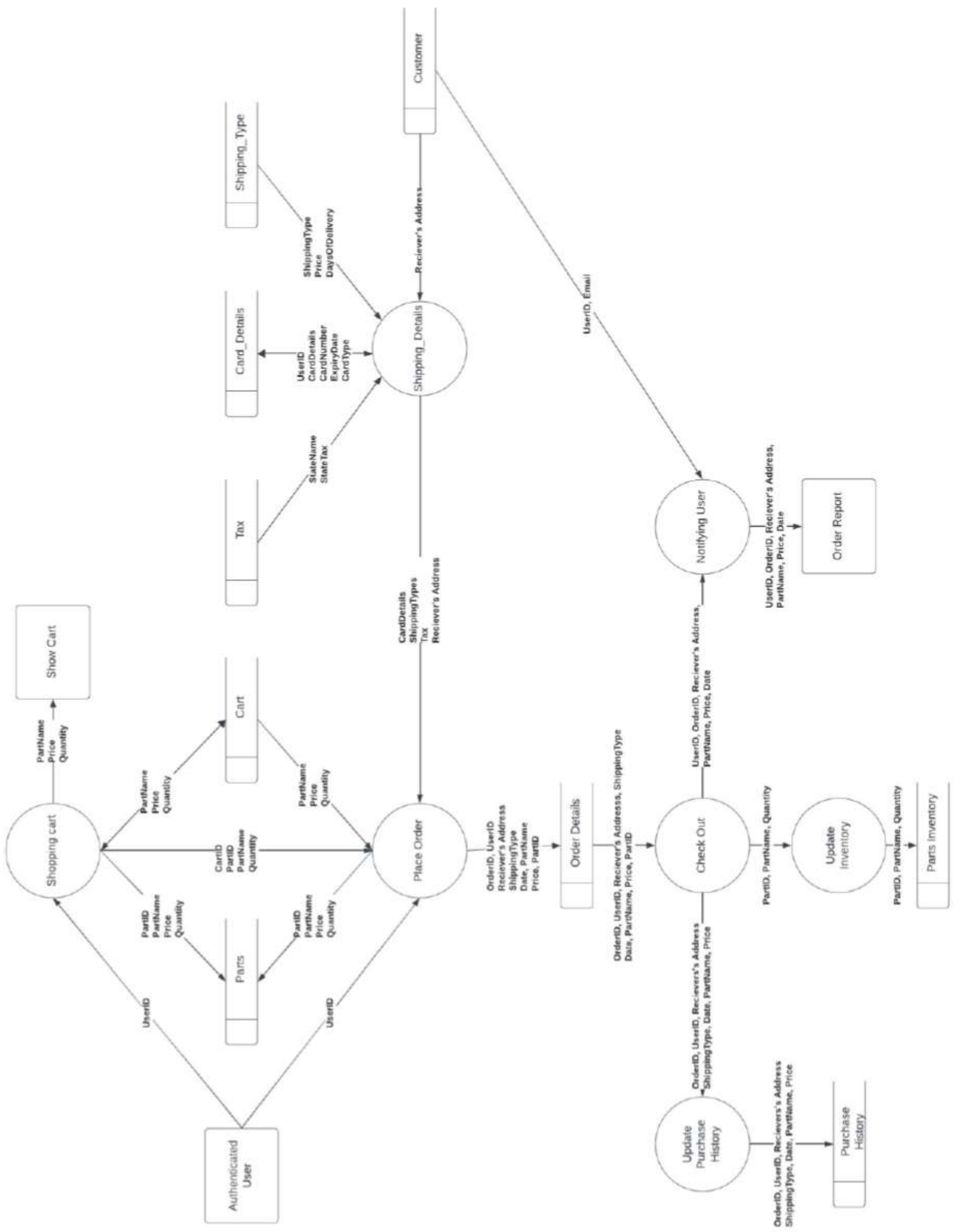
This DFD provides a comprehensive view of the flow of data through the customer purchase process, from authentication to order placement and inventory management.

- **Authenticated User**: This represents a user who has successfully logged in to the system.
- **Shopping Cart**: Users can add parts to their shopping cart, with details like part name, price, and quantity. They can also view the contents of the cart.
- **Parts**: This entity stores information about the car parts, including part ID, name, price, and quantity available.
- **Cart**: A temporary storage for the user's selected parts before purchase.
- **Place Order**: This process takes the parts from the cart and creates an order, associating it with the user's details and the chosen parts.
- **Order Details**: A record of each order placed, including order ID, user ID, shipping address, parts details, and price.
- **Tax**: Calculation of taxes based on the state associated with the receiver's address.
- **Card_Details**: Storage of the user's payment information.
- **Shipping_Type**: Details of the shipping method, including cost and estimated delivery days.
- **Shipping_Details**: Combines shipping type, card details, and tax information to process the order's shipping.
- **Customer**: The final recipient of the order, associated with the receiver's address.
- **Check Out**: The process where the order details are finalized, and the user proceeds to payment.
- **Notifying User**: Once the order is placed, the user is notified of the transaction details.
- **Order Report**: A detailed report of the order that can be used for confirmation and record-keeping.
- **Update Purchase History**: After an order is placed, the purchase history is updated with the order details.
- **Purchase History**: A log of all the purchases made by the user.
- **Update Inventory**: After an order is checked out, the parts inventory is updated to reflect the new stock levels.

VKUR Your Ideas, Our Innovation

Confidential: © 2023 VKUR Innovations. All rights reserved.

- **Parts Inventory**: The actual inventory of parts available for sale.

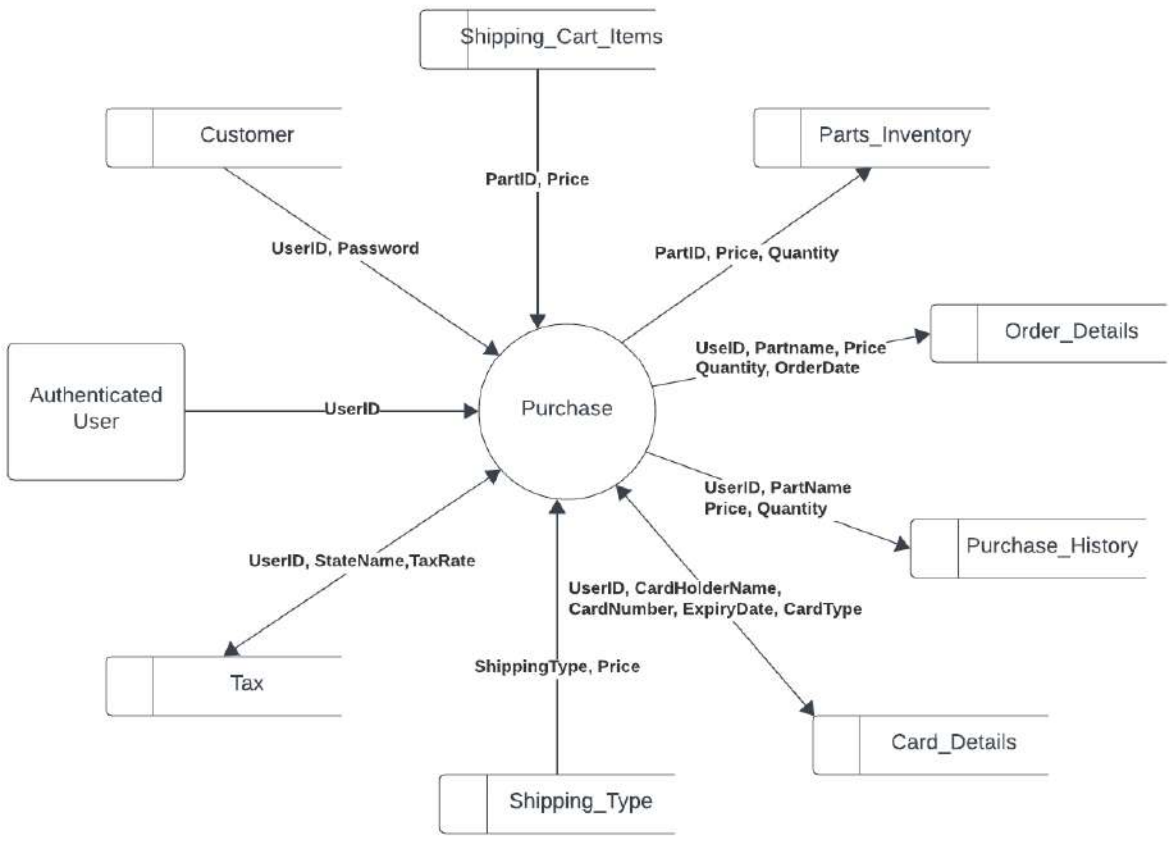

Customer
Shipping_Type
Card_Details
Tax
Shipping_Details
Recievei's Address
ShippingType Price DaysOfDelivery
UserID CardDetails CardNumber ExpiryDate CardType
StateName StateTax
UserID, Email
Notifying User
Order Report
UserID, OrderID, Reciever's Address, PartName, Price, Date
Show Cart
PartName Price Quantity
Shopping cart
Cart
CardDetails ShippingTypes Tax Reciever's Address
Place Order
CartID PartID PartName Quantity
PartID PartName Price Quantity
Parts
UserID
Authenticated User
OrderID, UserID Reciever's Address ShippingType Date, PartName Price, PartID
Order Details
OrderID, UserID, Reciever's Addresss, ShippingType Date, PartName, Price, PartID
Check Out
UserID, OrderID, Reciever's Address, PartName, Price, Date
PartID, PartName, Quantity
Update Inventory
Parts Inventory
OrderID, UserID, Recievers's Address ShippingType, Date, PartName, Price
Update Purchase History
Purchase History

6.3.4 Data for managing purchases

The data flows from the customer, through the process of authentication, selecting parts, and finally making a purchase. Tax rates are applied, payment details are processed, and the purchase is recorded in the user's history.

Confidential: © 2023 VKUR Innovations. All rights reserved.

- **Customer**: Initiates the process by attempting to make a purchase.
- **Authenticated User**: A verified customer on the platform identified by UserID, ready to make a purchase.
- **Purchase**: The central process where the authenticated user's purchase is processed.
- **Parts_Inventory**: The data store containing all available parts for sale, including their ID, price, and quantity.
- **Shipping_Cart_Items**: A temporary data store where the parts selected by the customer are held before purchase.
- **Order_Details**: Where the details of each order are recorded after a purchase is made.
- **Purchase_History**: A log of all purchases made by the user, recorded for historical reference.
- **Tax**: Contains tax information based on the user's location that is applied to the purchase.
- **Card_Details**: Holds the payment information of the user for processing the payment.
- **Shipping_Type**: Contains details about the shipping options available, used to calculate shipping charges and manage delivery.



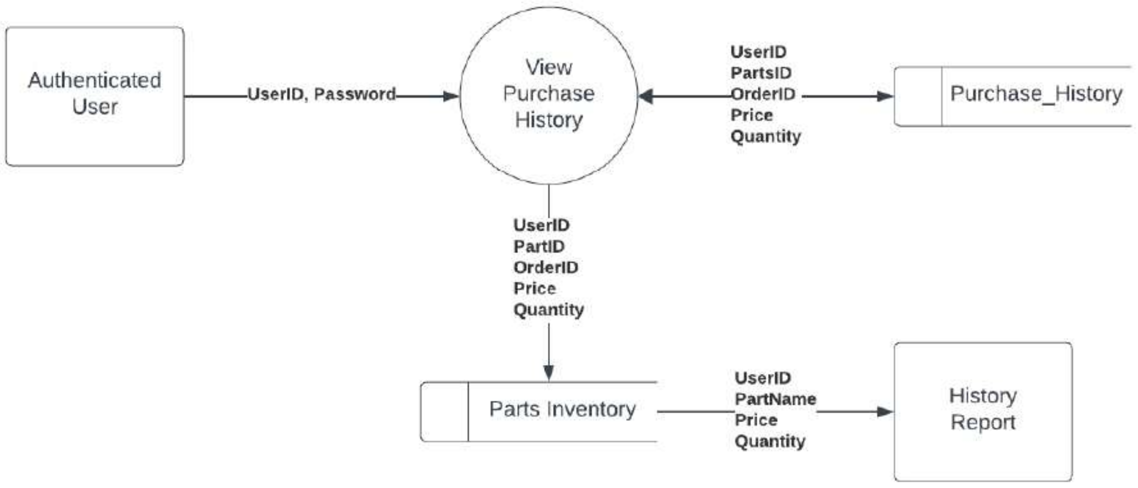
6.3.5 Purchase history and parts inventory

This DFD details how the system retrieves data from the purchase history and parts inventory to provide the user with a comprehensive report of their past transactions.

- **Authenticated User**: The user of the system who has been authenticated, typically after providing their UserID and Password.
- **View Purchase History**: The process that retrieves and displays the user's purchase history.
- **Purchase_History**: The data store that contains the records of all purchases made by the user, including details like UserID, PartsID, OrderID, Price, and Quantity.

Confidential: © 2023 VKUR Innovations. All rights reserved.

- **Parts Inventory**: This data store contains information about the parts that have been purchased, likely used to validate the parts that appear in the purchase history.
- **History Report**: A report generated from the purchase history data, which might include the part name, price, and quantity for the user to review.



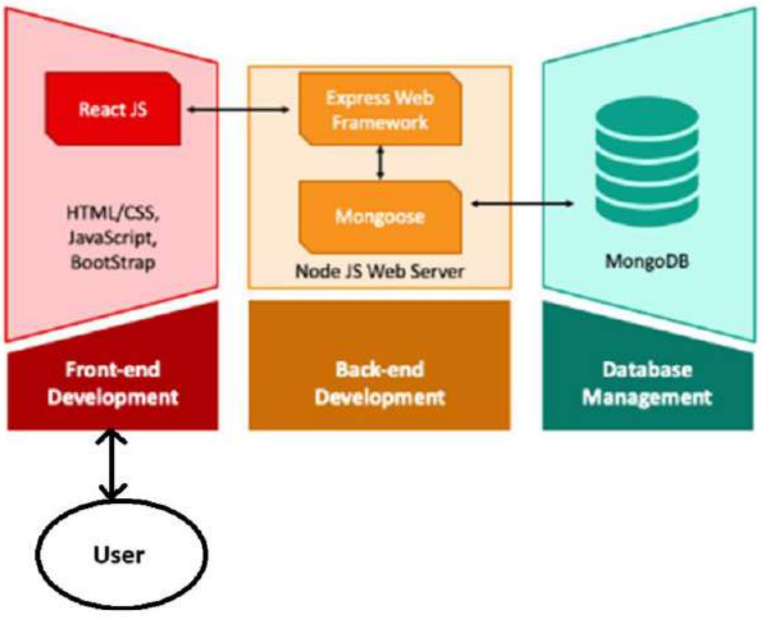
7. Deployment Strategy

7.1 Deployment Plan and Platforms

Deployment Plan:

- The platform is containerized using Docker, which encapsulates the application and its environment for consistency across different development and production stages.

VKUR
Your Ideas, Our Innovation



Confidential: © 2023 VKUR Innovations. All rights reserved.

- Continuous Integration/Continuous Deployment (CI/CD) pipelines are established to automate the testing and deployment process. Tools like Jenkins or GitHub Actions will be used for this purpose.
- The application is deployed to AWS cloud services which offer scalable infrastructure and managed services like databases and storage.

Deployment Platforms:

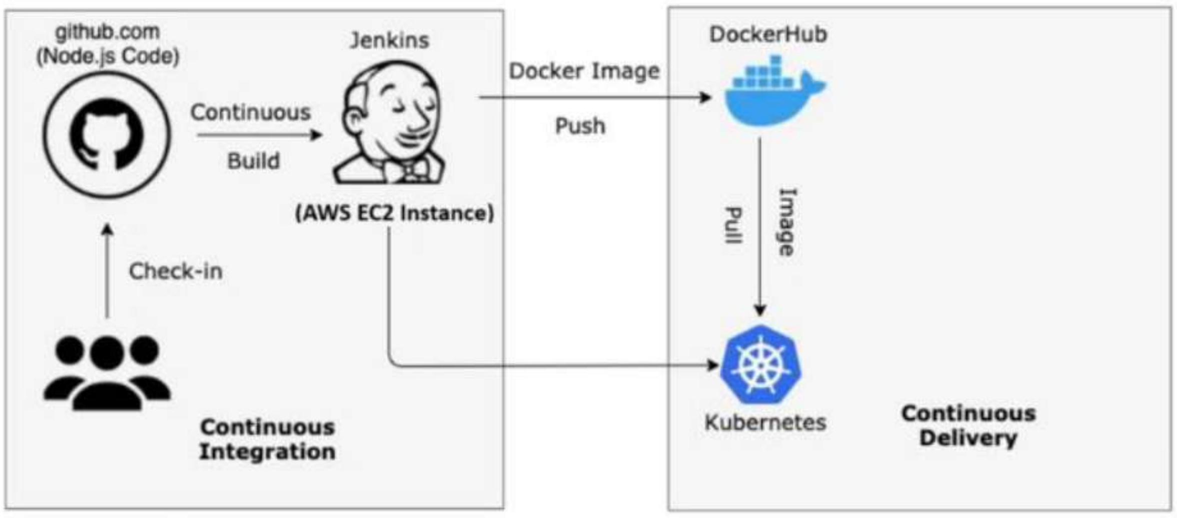
- Development Environment: A staging version is deployed internally for testing and QA to ensure functionality works as expected before public release.
- Production Environment: The production version is deployed to a live server. Load balancers and redundancy mechanisms are put in place to ensure high availability and fault tolerance.
- Monitoring Tools: Prometheus is implemented to monitor the application's performance and health in real-time.

7.2 Continuous Integration/Continuous Deployment (CI/CD) Pipeline

CI/CD pipeline ensures that "Don't Junk It - List It" can rapidly and reliably deliver updates and new features to users.

- **Code Repository (GitHub.com):** Developers push the Node.js code to GitHub. Each commit triggers the CI/CD process.
- **Continuous Integration (Jenkins):** Jenkins, set up on an AWS EC2 instance, automatically detects the new commit (check-in) and initiates the continuous build process. Jenkins compiles the code and runs automated tests to ensure code quality and functionality.
- **Docker Image Creation:** After a successful build and test process, Jenkins creates a Docker image of the application, encapsulating the application and its environment.
- **Image Repository (DockerHub):** The Docker image is then pushed to DockerHub, a service for managing Docker images.
- **Kubernetes:** The orchestration platform, Kubernetes (not specifically GKE in this case), pulls the latest Docker image from DockerHub.
- **Continuous Delivery:** Kubernetes handles the deployment of the Docker image across the appropriate clusters. It manages the application's scaling, self-healing, and updates.

VKUR
Your Ideas, Our Innovation



Confidential: © 2023 VKUR Innovations. All rights reserved.

8. Maintenance and Support

8.1 Maintenance Plan:

- **Regular Updates**: We'll update the website often to make sure everything works well and safely.
- **Watching Over Performance**: We will use special tools to keep an eye on how well the website is working and fix any problems before they become big issues.
- **Taking Care of Data**: We'll make sure all the data on the website is backed up and safe.
- **Listening to Users**: We want to know what users think so we can make the website better over time.

8.2 Support Procedures and Channels:

- **Helpdesk**: We have a friendly team ready to answer questions or solve problems, which you can reach by email, chat, or phone.
- **Online Help**: There's a section on the website with common questions and answers, guides, and a place for users to talk and help each other.
- **Quick Help for Big Problems**: If something really bad happens to the website, we have a special plan to fix it quickly.
- **Training for Support Team**: We make sure our support team knows all about the latest updates and how to help users in the best way.

9. Conclusion

We are confident in the promise and utility that the "Don't Junk It - List It" platform offers. It is poised to significantly enhance the efficiency and effectiveness with which users can buy and sell used car parts. The platform's architecture, consisting of a user-friendly front end, a robust back end, and a secure database, ensures simplicity in design and ease of use. As the project transitions to the hands of our client for ongoing management, the straightforward and streamlined design assures them a product that is not only beneficial but also manageable. We anticipate that this platform will markedly boost their operational productivity and user engagement.

VKUR
Your Ideas, Our Innovation

Confidential: © 2023 VKUR Innovations. All rights reserved.